Comparison UUnifast and DRS

Souvik Sarkar and Mario Günzel

September 26, 2022

Abstract:
This is a short comparison of the evaluation results obtained from UUnifast and from DRS.

UUniFast:

For UUniFast suspension time ['sslength'] is drawn uniformly from the interval between the minimum suspension length value and maximum suspension length value. We have the following three setups:

- Setup 1 Short Suspension [0.0(Ti - Ci), 0.2(Ti - Ci)]
- Setup 2 Moderate Suspension [0.2(Ti - Ci), 0.4(Ti - Ci)]
- Setup 3 Long Suspension [0.4(Ti - Ci), 0.6(Ti - Ci)]

DRS:

Unlike UUniFast, Dirichlet- Rescaling Algorithm are used for asymmetric constraints and works with separate upper bounds and lower bounds for each task.

The three different setups for DRS used here:

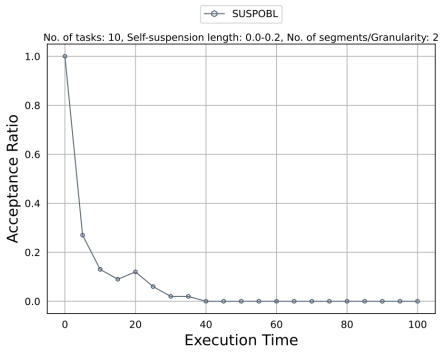
- Setup 1 (minsus+ex=0.1*number of tasks per set, maxsus+ex=1.0)
- Setup 2 (minsus+ex=0.3*number of tasks per set, maxsus+ex=1.0)
- Setup 3 (minsus+ex=0.5*number of tasks per set, maxsus+ex=1.0)

Here, we are taking three different setups each with different execution + suspension time but same execution time.

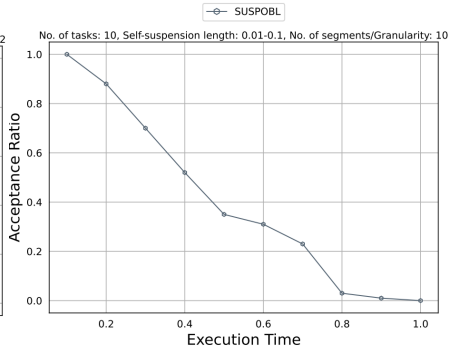
1 Suspension Oblivious

We are going to generate a suspension-oblivious schedule for the DRS and UUniFast setups explained above

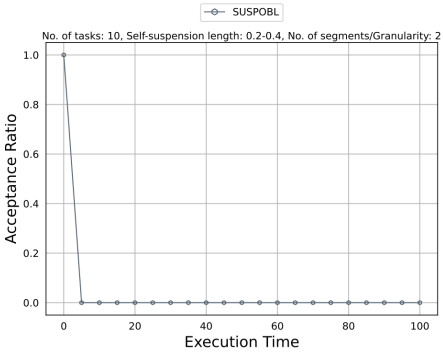
(UUnifast) **(DRS)**



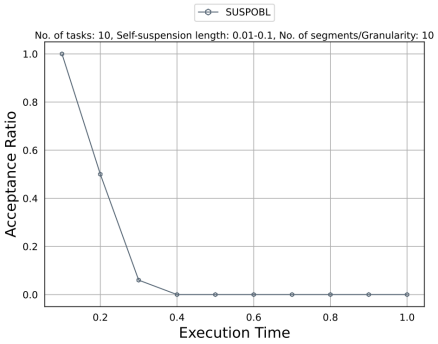
First UUnifast setup.



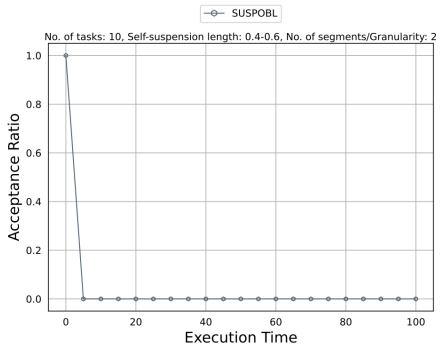
First DRS setup.



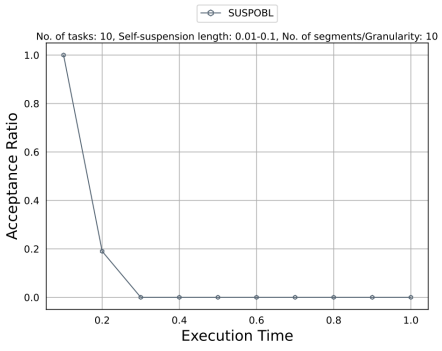
Second UUnifast setup.



Second DRS setup.



Third UUnifast setup.

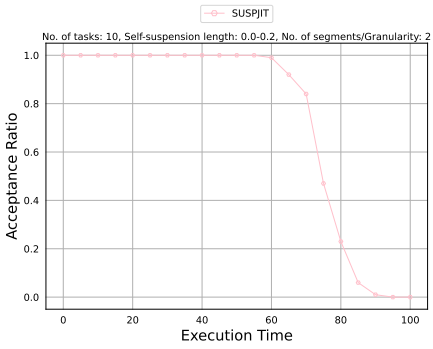


Third DRS setup.

2 Suspension Jitter

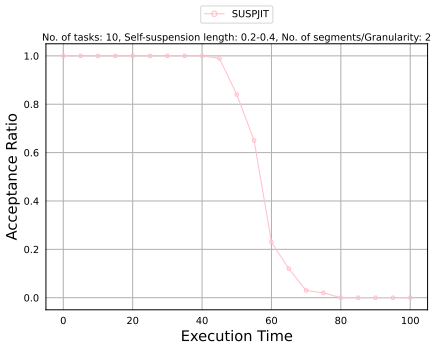
We are going to generate a suspension-jitter schedule for the DRS and UUniFast setups explained above

(UUnifast) **(DRS)**



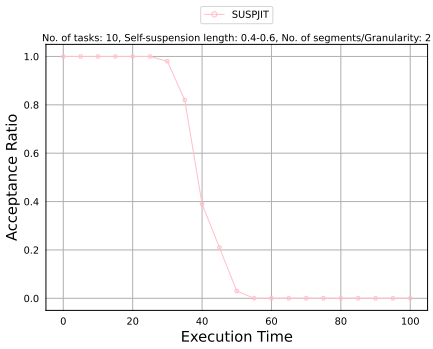
First UUnifast setup.

First DRS setup.

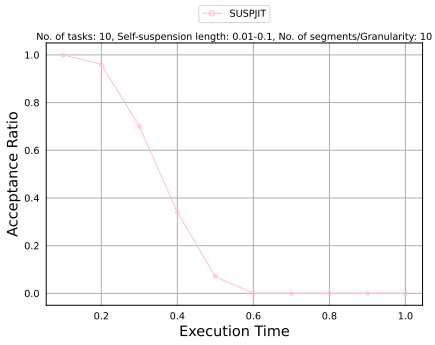


Second UUnifast setup.

Second DRS setup.



Third UUnifast setup.



Third DRS setup.

Evaluation:

So, the parameter we use to comapare is **Acceptance Ratio** with respect to the execution time i.e the percentage of tasksets that got accepted for a particular execution time. We are generating 100 task sets per configuration with 10 tasks per set with a suspension values ranging between 0,0 - 0,6.

We use two methods DRS and UUniFast to generate a set of utilization values with motive of generating (Usum, ubound, lbound) and (Usum) respectively. The resulted tasksets were then tested under **Suspension Oblivious** and **Suspension Jitter**. We see in the resulting graphs (above) that Acceptance Ratio gets reduced for tasksets with higher suspension intervals.

Using Suspension oblivious methods we see a higher acceptance ratio for taskets generated using DRS over UUniFast method. But, we also see that those same DRS task sets acheive a slightly lower acceptance ratio using Suspension Jitter schedule.